

POSTED WRITE BUFFERS AND METHOD OF POSTING WRITE REQUESTS IN
MEMORY MODULES

TECHNICAL FIELD

This invention relates to computer systems, and, more particularly, to a
5 computer system having a processor or other memory access device coupled to a
plurality of memory modules each of which includes a memory hub coupled to a
plurality of memory devices.

BACKGROUND OF THE INVENTION

Computer systems use memory devices, such as dynamic random access
10 memory (“DRAM”) devices, to store data that are accessed by a processor. These
memory devices are normally used as system memory in a computer system. In a
typical computer system, the processor communicates with the system memory through
a processor bus and a memory controller. The processor issues a memory request,
which includes a memory command, such as a read command, and an address
15 designating the location from which data or instructions are to be read. The memory
controller uses the command and address to generate appropriate command signals as
well as row and column addresses, which are applied to the system memory. In
response to the commands and addresses, data are transferred between the system
memory and the processor. The memory controller is often part of a system controller,
20 which also includes bus bridge circuitry for coupling the processor bus to an expansion
bus, such as a PCI bus.

Although the operating speed of memory devices has continuously
increased, this increase in operating speed has not kept pace with increases in the
operating speed of processors. Even slower has been the increase in operating speed of
25 memory controllers coupling processors to memory devices. The relatively slow speed
of memory controllers and memory devices limits the data bandwidth between the
processor and the memory devices.

In addition to the limited bandwidth between processors and system memory devices, the performance of computer systems is also limited by latency problems that increase the time required to read data from system memory devices. More specifically, when a memory device read command is coupled to a system 5 memory device, such as a synchronous DRAM (“SDRAM”) device, the read data are output from the SDRAM device only after a delay of several clock periods. Therefore, although SDRAM devices can synchronously output burst data at a high data rate, the delay in initially providing the data can significantly slow the operating speed of a computer system using such SDRAM devices as system memory.

10 Memory read latency may also be adversely impacted by the need to write data to memory devices. More specifically, if a controller issues a write request followed by a read request, it may not be possible for a memory device to which the requests are issued to respond to the read request until after the write request has been serviced. The memory read latency will therefore be increased by the time required to 15 service the write request. Therefore, write requests can considerably increase memory read latencies.

One approach to limiting the degree to which write requests can increase memory read latency is to use posted write buffers to store write requests while a read request is being serviced. In a computer system having a posted write buffer, the 20 processor or other memory access device can issue a write request even if the memory device to which the write request is directed is busy servicing a prior write or read request. Using this approach, memory requests can be serviced out of order since an earlier write request can be stored in the posted write buffer while a subsequent read request is being serviced. The ability to buffer write requests to allow a read request to 25 be serviced can greatly reduce memory read latency since read requests can be given first priority regardless of their chronological order.

The use of a posted write buffer can provide advantages in addition to reducing memory read latency. For example, a series of write requests interspersed with read requests can be stored in the posted write buffer to allow the read requests to be 30 serviced in a pipelined manner followed by servicing the stored write requests in a

pipelined manner. Accumulating write requests in this manner also tends to avoid placing alternating write and read requests on a memory bus, which can require that lengthy settling times be provided between coupling the write request to the memory device and subsequently coupling the read request to the memory device.

5 Although the use of posted write buffers provides significant advantages in conventional computer systems, it is likely to be less advantageous in a computer system having a memory system using a hub architecture. In a conventional computer system, the posted write buffer is normally a part of the system controller or the processor. A posted write buffer in the processor or controller can adequately handle
10 the write requests that a processor issues to several memory devices. In a hub architecture, a processor is coupled to several memory modules through a system controller or similar device. Each of the memory modules includes a memory hub coupled to the controller and to several memory devices that are also part of the memory module. A posted write buffer located in the controller is likely to be inadequate in
15 handling the vastly higher rate of write requests that would be directed to several memory modules each of which includes a memory hub coupled to several memory devices. Not only is the bandwidth that the posted write buffer would be required to handle vastly greater with a hub architecture, but the difficulty in ensuring write buffer coherency is also vastly greater. More specifically, the posted write buffer must be able
20 to handle a “read around write” situation in which a read request to a memory address is processed prior to an earlier occurring write request to the same memory address. Otherwise, the read request will return the wrong data because the write request, which would have stored the correct data at that memory address, has not yet been serviced. The large amount of write requests that would need to be buffered with a hub
25 architecture in a system having a large number of memory addresses would make it very difficult to ensure coherency in conventional posted write buffers.

There is therefore a need for an architecture that provides the advantages of a posted write buffer in a computer system or other electronic system using a memory hub architecture, thereby providing a memory system having a high bandwidth and low
30 latency.

SUMMARY OF THE INVENTION

A memory module that may be used in a computer system includes a memory hub coupled to a plurality of memory devices. The memory hub includes a link interface receiving memory requests for access to at least one of the memory devices, and a memory device interface coupled to the memory devices. The memory device interface couples memory requests to the memory devices for access to at least one of the memory devices and receives read data responsive to at least some of the memory requests. The memory hub also includes a posted write buffer coupled to the link interface and the memory device interface. The posted write buffer stores write memory requests and subsequently couples the write memory requests to the memory device interface. A read request path in the memory hub, which may include a memory sequencer, couples read memory requests from the link interface to the memory device interface and couples read data from the memory device interface to the link interface. The posted write buffer may also include coherency circuitry for determining if a read request is directed to an address to which a write request has been posted. If so, the read data responsive to the read request is provided from the posted write buffer rather from one of the memory devices.

BRIEF DESCRIPTION OF THE DRAWINGS

20 Figure 1 is a block diagram of a computer system according to one example of the invention in which a memory hub is included in each of a plurality of memory modules.

25 Figure 2 is a block diagram of a memory hub used in the computer system of Figure 1, which contains a posted write buffer according to one example of the invention.

Figure 3 is a flow chart showing a process for controlling the number of posted write requests that have been accumulated in a posted write buffer in the memory hub shown in Figure 2.

DETAILED DESCRIPTION OF THE INVENTION

A computer system 100 according to one example of the invention is shown in Figure 1. The computer system 100 includes a processor 104 for performing various computing functions, such as executing specific software to perform specific calculations or tasks. The processor 104 includes a processor bus 106 that normally includes an address bus, a control bus, and a data bus. The processor bus 106 is typically coupled to cache memory 108, which, as previously mentioned, is usually static random access memory (“SRAM”). Finally, the processor bus 106 is coupled to a system controller 110, which is also sometimes referred to as a “North Bridge” or 10 “memory controller.”

The system controller 110 serves as a communications path to the processor 104 for a variety of other components. More specifically, the system controller 110 includes a graphics port that is typically coupled to a graphics controller 112, which is, in turn, coupled to a video terminal 114. The system controller 110 is 15 also coupled to one or more input devices 118, such as a keyboard or a mouse, to allow an operator to interface with the computer system 100. Typically, the computer system 100 also includes one or more output devices 120, such as a printer, coupled to the processor 104 through the system controller 110. One or more data storage devices 124 are also typically coupled to the processor 104 through the system controller 110 to 20 allow the processor 104 to store data or retrieve data from internal or external storage media (not shown). Examples of typical storage devices 124 include hard and floppy disks, tape cassettes, and compact disk read-only memories (CD-ROMs).

The system controller 110 is coupled to several memory modules 130a,b,...n, which serve as system memory for the computer system 100. The memory 25 modules 130 are preferably coupled to the system controller 110 through a high-speed link 134, which may be an optical or electrical communication path or some other type of communications path. In the event the high-speed link 134 is implemented as an optical communication path, the optical communication path may be in the form of one or more optical fibers, for example. In such case, the system controller 110 and the 30 memory modules will include an optical input/output port or separate input and output

ports coupled to the optical communication path. The memory modules 130 are shown coupled to the system controller 110 in a multi-drop arrangement in which the single high-speed link 134 is coupled to all of the memory modules 130. However, it will be understood that other topologies may also be used, such as a point-to-point coupling 5 arrangement in which a separate high-speed link (not shown) is used to couple each of the memory modules 130 to the system controller 110. A switching topology may also be used in which the system controller 110 is selectively coupled to each of the memory modules 130 through a switch (not shown). Other topologies that may be used will be apparent to one skilled in the art.

10 Each of the memory modules 130 includes a memory hub 140 for controlling access to 6 memory devices 148, which, in the example illustrated in Figure 2, are synchronous dynamic random access memory (“SDRAM”) devices. However, a fewer or greater number of memory devices 148 may be used, and memory devices other than SDRAM devices may, of course, also be used. The memory hub 140 is 15 coupled to each of the system memory devices 148 through a bus system 150, which normally includes a control bus, an address bus and a data bus.

One example of the memory hub 140 of Figure 1 is shown in Figure 2. The memory hub 140 includes a link interface 152 that is coupled to the high-speed link 134. The nature of the link interface 152 will depend upon the characteristics of the 20 high-speed link 134. For example, in the event the high-speed link 134 is implemented using an optical communications path, the link interface 152 will include an optical input/output port or separate input and output ports and will convert optical signals received through the optical communications path into electrical signals and electrical signals into optical signals that are transmitted to the optical communications path. In 25 any case, the link interface 152 may include a variety of conventional interface circuitry such as, for example, a first-in, first-out buffer (not shown), for receiving and storing memory requests as they are received through the high-speed link 134. The memory requests can then be stored in the link interface until they can be processed by the memory hub 140.

A memory request received by the link interface 152 is processed by first transferring the request to a posted write buffer 160. If the memory request is a write request, the request will be stored for subsequent processing, as explained in greater detail below. If the memory request is a read request, conventional coherency circuitry

5 162 in the posted write buffer 160 checks to determine if the read request is to an address to which a previous write request still stored in the buffer is directed. For example, the coherency circuitry can 162 post write request addresses to a posted address buffer. The address forming part of each read request can then be compared to the addresses in the posted address buffer.

10 In the event of an address match, the read data called for by the read request are returned from the posted write buffer 160. The posted write data are coupled from the posted write buffer 160 to one port of a multiplexer 164. The coherency circuitry 162 in the posted write buffer 160 also applies a control signal to the multiplexer 164 to couple the posted write data to the link interface 152. In the

15 coherency circuitry 162 does not detect an event match, the posted write buffer 160 applies an active high MISS signal to a memory sequencer 170, which receives the read request from the link interface 152 as "Read Around Write" request. The memory sequencer 170 responds to the MISS signal by placing the read request in the proper sequence with other read requests and subsequently coupling the read request to a

20 memory device interface 174. The memory sequencer 170 may also convert the read requests from the format output from the system controller 110 (Figure 1) into a read request having a format that can be used by the memory devices 148. These re-formatted request signals will normally include memory command signals, which are derived from memory commands contained in the memory request received by the

25 memory hub 140, and row and column address signals, which are derived from an address contained in the memory request received by the memory hub 140. For example, where the memory devices 148 are conventional DRAM devices, the memory sequencer 170 will output row address signals, a row address strobe ("RAS") signal, an active low write/active low read signal ("W/R*"), column address signals and a column

30 address strobe ("CAS") signal. The re-formatted memory requests are preferably output

from the memory sequencer 170 in the order they will be used by the memory devices 148.

After the memory device interface 174 has applied the read request to the memory devices 148, the memory devices 148 return read data called for by the request 5 to the memory device interface 174. The memory device interface 174 then couples the read data to the other input port of the multiplexer 164. If the posted write buffer 160 outputs an inactive low MISS signal, it also couples a control signal to the multiplexer 164 that causes the read data from the memory device interface 174 to be coupled to the link interface 152. The link interface 152 then couples the read data to the controller 10 110 through the high-speed link 134.

As an alternative to reading from the posted write buffer 160 in the event its internal coherency circuitry 162 detects a match between a memory read address and a posted write address, the coherency circuit 162 may instead cause the read request to be delayed and immediately issue the posted write request to write data to the memory 15 devices 148. The memory hub 140 can then issue the read request to the memory devices 148 to read the data that has now been written to the memory devices 148 at the memory read address. Other means of ensuring coherency that may be used in the event a read request is serviced before servicing a write request to the same address will be apparent to one skilled in the art.

20 As mentioned above, the memory sequencer 170 applies the read around write memory requests to the memory device interface 174. The nature of the memory device interface 174 will again depend upon the characteristics of the memory devices 148. In any case, the memory device interface 174, like the link interface 152, may include a FIFO buffer (not shown), for receiving and storing one or more memory 25 requests as they are received from the link interface 152. The memory request can be stored in the FIFO buffer until they can be processed by the memory devices 148. Alternatively, the memory device interface 174 can simply pass the memory requests to the memory devices 148. In the event the memory device interface 174 stores several memory requests until they can be processed by the memory devices 148, the memory

device interface 174 may re-order the memory requests so that they are applied to the memory devices 148 in some other order.

At some point during the operation of the memory hub 140, write requests posted in the buffer 160 will accumulate to a level that they must be serviced.

- 5 Various techniques for controlling the level of posted write requests that can accumulate will be discussed below. In any case, when the posted write requests have accumulated to a sufficient level, the posted write buffer 160 couples the write requests to the memory device interface 174. These write requests will normally include the write data as well as the address to which that data are to be written and appropriate command
- 10 signals, as are well known to one skilled in the art. The memory device interface 174 then passes the write requests on to the memory device 148 addressed by the write request.

The accumulation of posted write requests in the buffer 160 may be controlled by the process 200 shown in Figure 3. The process 200 is entered at 202 and

- 15 waits at 206 for a new write request to be received from the link interface 152. When a new write request is received from the link interface 152, a counter keeping track of the number W of posted write requests accumulated in the posted write buffer 160 is incremented at 210. The process then checks at 214 to determine if the number of posted write requests W accumulated exceeds a maximum number, W_{MAX} . If W
- 20 exceeds W_{MAX} , the posted write buffer 160 couples a write request to the memory device interface 174 at 218. The posted write buffer 160 can issue the write requests in any desired order, such as on a first-in, first-out order, a first-in, last-out order or some or other order, as is well known to one skilled in the art. In any case, once the posted write buffer 160 has issued the write request, the number of posted write requests W
- 25 accumulated is decremented by one at 220. The process then returns to 206 to await another write request.

Using the procedure 200 described above, the posted write buffer 160 will issue write requests only when W_{MAX} write requests have been accumulated in the buffer 160 until the number of write requests posted in the buffer 160 falls to some

- 30 predetermined number which may be zero. Under these circumstances, the posted write

buffer 160 would never issue the final W_{MAX} write requests. For this reason, if the process 200 determines at 214 that W does not exceed W_{MAX} , the process 200 checks at 226 to determine if the time T_W any write request has remained in the posted write buffer 160 exceeds a maximum time T_{MAX} . If so, the process 200 branches to 218 to 5 issue the overdue write requests, as previously explained. Otherwise, the process returns to 206 to await a new write request.

Although a specific example of a technique for controlling the accumulation of posted write requests has been explained with reference to Figure 3, it will be apparent to one skilled in the art than other techniques may be used. For 10 example, posted write requests could be issued by the buffer 160 after they had remained there for a predetermined period of time regardless of how many write requests had been accumulated. The posted write buffer 160 can also issue the posted write requests whenever read requests are not being processed by the memory hub 140 or the memory devices 148 coupled thereto. Also, the values W_{MAX} and T_{MAX} may be 15 adjusted depending on various operating conditions. For example, the maximum number of posted write requests accumulated W_{MAX} could be adjusted depending upon the rate at which posted write requests are being accumulated. Other variations will be apparent to one skilled in the art.

From the foregoing it will be appreciated that, although specific 20 embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.